Making-off Patch FR DQH2

Avant-propos

Le modding de jeux vidéo constitue une violation des conditions d'utilisations. Et donc s'il est toléré dans la plupart des cas, il n'est pas toujours possible d'agir en parfaite légalité ni de respecter strictement les droits d'auteurs, notamment lors de la publication d'un patch.

Un exemple flagrant en ce qui concerne la PS Vita est que nous devons avoir une console hackée pour profiter d'un patch de traduction.

Ainsi, la question est de savoir si nous agissons de façon légitime :

- Cracker sa console pour permettre de lancer son jeu favori en français
- Enfreindre un copyright pour publier une portion du jeu améliorée
- Utiliser le travail d'autres contributeurs en toute transparence
- Ne pas gagner d'argent sur le dos de qui que ce soit

Tout cela me paraît juste, et donc c'est avec la conscience tranquille que j'ai parfois dévié du droit chemin vertueux, pour permettre les différents patchs FR jusqu'à maintenant !

Sur ce, je vous laisse apprécier (ou pas) le déroulement de ce patch décrit dans l'ordre presque chronologique, d'une manière simplifiée pour la meilleure compréhension de tous !

Préparation des fichiers à traduire

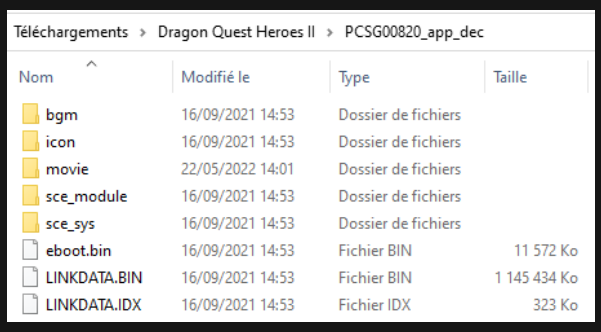
Dans le cas des jeux PS Vita, le contenu est sous forme cryptée et il faut une clef appelée ZRif pour les déchiffrer. Même si cela permet le piratage de copies déchiffrées, obtenir cette clef sur NoPayStation est une étape primordiale pour pouvoir modifier les fichiers.

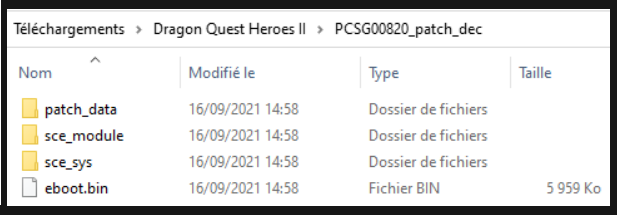
Sur la PS Vita, quand on a le jeu installé, les fichiers de **Dragon Quest Heroes II** sont dans :

- ux0:/app/PCSG00820 pour le jeu principal
- ux0:/patch/PCSG00820 pour l'update 1.06

PSVpfsparser (https://github.com/motoharu-gosuto/psvpfstools) effectue l'opération de décryptage avec la ligne de commande sur chaque dossier : `psvpfsparser.exe -i "app\PCSG00042" -o "C:\Output" -z <Zrif> -f http://cma.henkaku.xyz/`

On obtient les fichiers de la PSVita, déchiffrés et donc exploitables.





Un dernier préparatif est nécessaire sur le eboot.bin (format FSELF) pour le transformer en ELF et pouvoir le modder. Sachant que le eboot.bin de l'update 1.06 remplace celui d'origine, c'est ce fichier qu'il faut sélectionner avec **FAGdec** sur la PSVita. On conserve ainsi un "eboot.bin" v1.06 décrypté (format Sony) et le ELF décrypté, lisible et modifiable qu'on peut renommer "eboot.bin.elf"

En plus, on aura la possibilité de le décompiler avec Ghidra si besoin en suivant ces instructions : https://forum.devchroma.nl/index.php?topic=88.0

Obtention des assets traduits

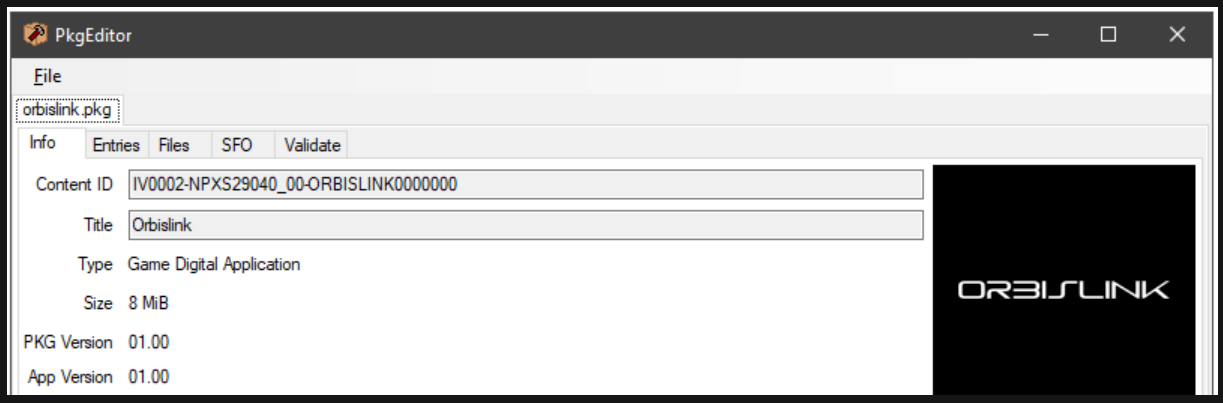
Les versions PS4 et PC possèdent une traduction officielle en Français, ce qui n'est pas le cas de la version Vita (JP only). Le 1er objectif est donc d'en récupérer le contenu traduit, dit « assets ».

On trouve en ligne le fichier **EP0082-CUSA06740_00-DQHGAME000000000-A0100-V0106.pkg** (21 Gb)

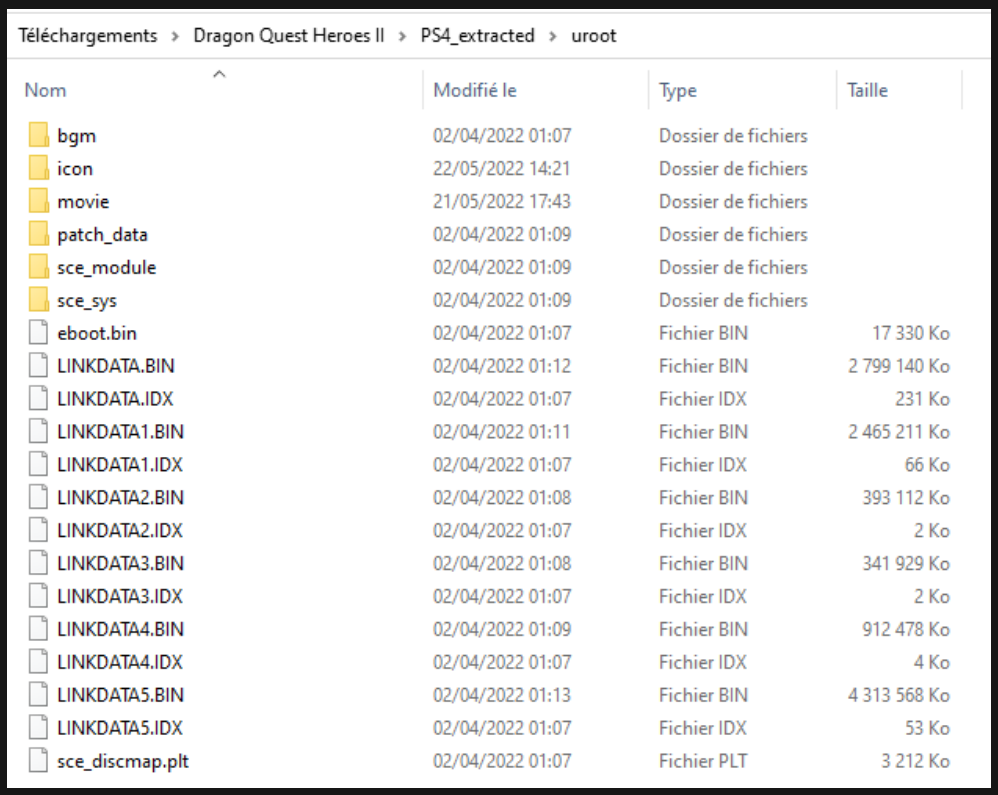
La librairie LibOrbis va permettre d'en extraire le contenu :

https://github.com/maxton/LibOrbisPkg/tree/master

Ces utilitaires sont livrés avec une GUI nommée « PKGeditor »



Mais seule la commande `pkg_extract <input.pkg> <output_directory>` est nécessaire pour obtenir :



Il est intéressant de noter la similarité avec la PS Vita, et notamment :

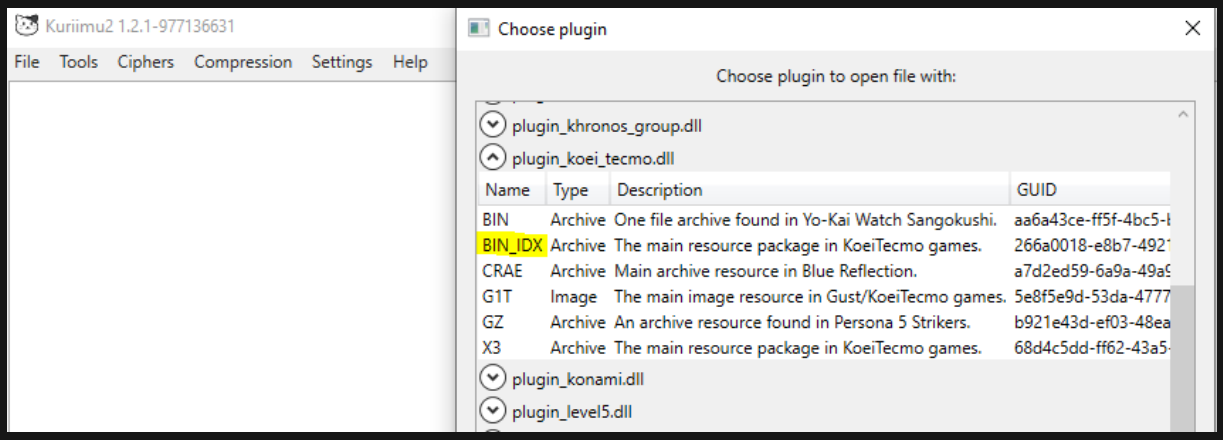
- Le dossier « movie » qui va nous permettre de sous-titrer les vidéos qui sont en MP4
- Les LINKDATA.BIN et .IDX qu'on retrouve dans app/PCSG00820
- Le dossier « patch_data » qu'on retrouve dans patch/PCSG00820.
- Et enfin le eboot.bin.

Extraction des assets

Le LINKDATA de la PS4 n'est pas exploitable par la PSVita pour plein de raisons techniques, dont l'une est facilement compréhensible : sa résolution est inférieure (960x544 px) et donc les textures doivent être converties. Le fichier binaire (BIN) est accompagné d'un fichier index (IDX) qui le décrit.

Il faut donc réaliser une extraction de ces fichiers, "unpack". Tout en gardant à l'esprit qu'il faudra être capable de réinsérer ensuite, "repack".

Le 1er réflexe est de vérifier si Kuriimu ou Kuriimu2 sont capables de traiter ce genre de fichier, car on aurait les deux fonctions exporter et importer.

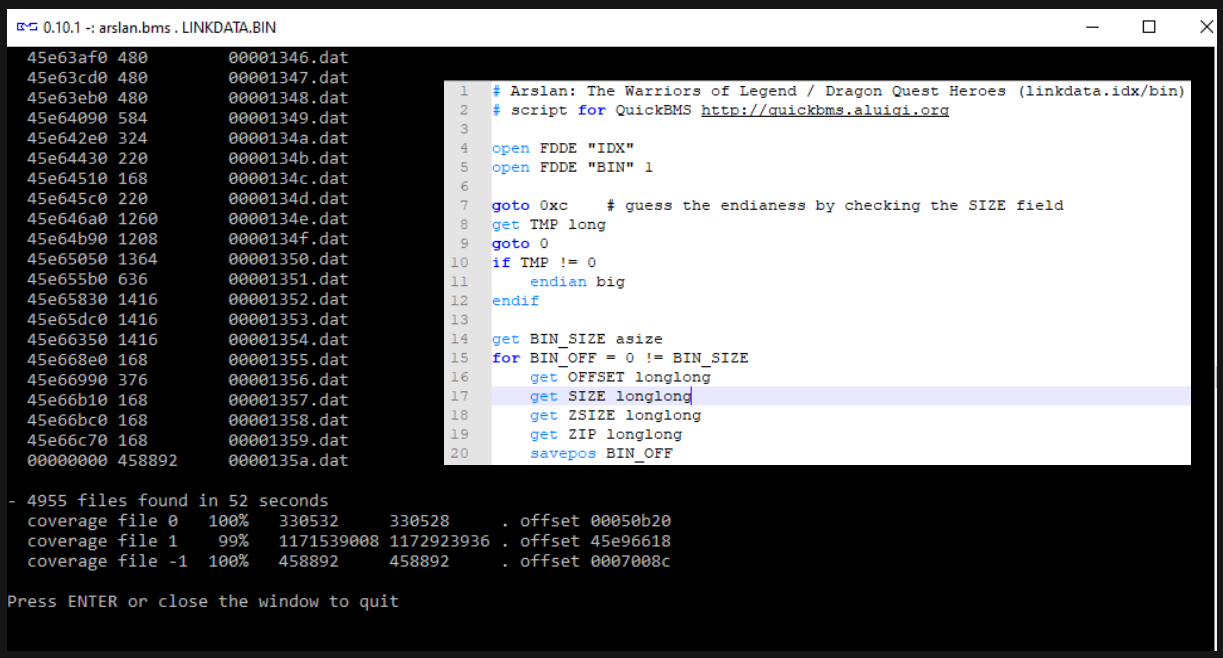


Hélas il y a bien un plugin pour les BIN/IDX mais il ne gère pas DQH2 (ni PSVita, ni PS4)

Donc le 2ème réflexe est d'aller fouiller dans le temple du romhacking : https://zenhax.com/index.php

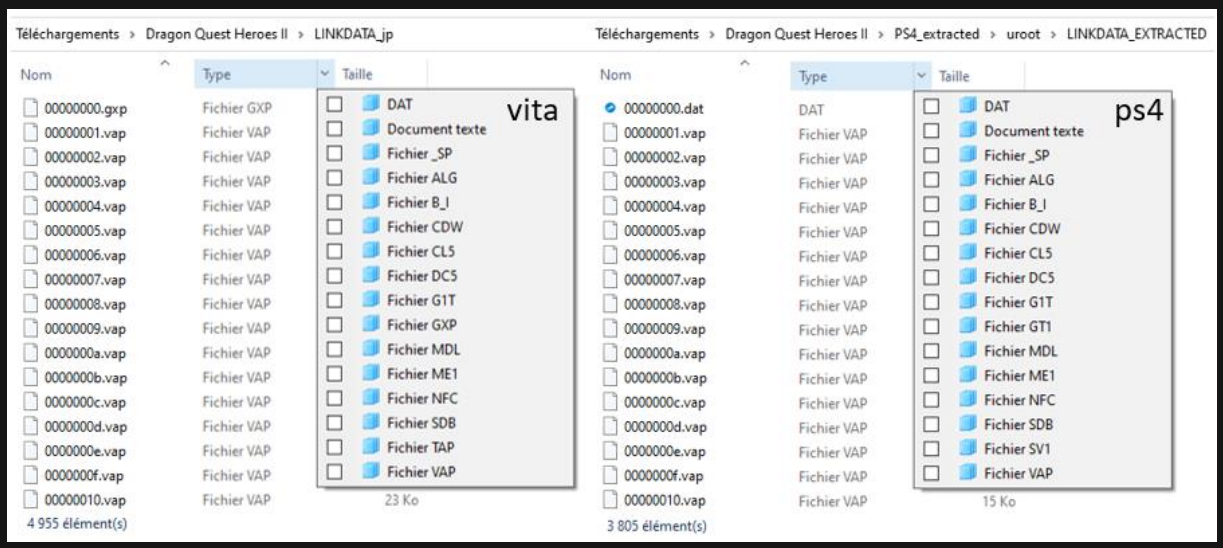
Et notamment parmi les scripts QuickBMS https://aluigi.altervista.org/quickbms.htm

Il s'avère que le script **arslan.bms** permet d'extraire et décompresser parfaitement tout le contenu des LINKDATA ! L'inconvénient est qu'il n'y a pas de méthode pour réimporter ensuite, et inverser les opérations du script n'est pas une chose aisée.



Tri des fichiers

Une fois l'extraction accomplie sur les 2 versions, celle de la PS Vita et celle de la PS4, on se retrouve avec des structures très similaires, sans les noms de fichiers n'ont pas été conservé lorsque les assets ont été empaquetés en BIN/IDX :



Les extensions détectées dans les headers sont les mêmes, il y a moins de fichiers sur la version PS4 mais c'est dû au fait que seul le LINKDATA principal a été extrait, pas les autres numérotés de 1 à 5.

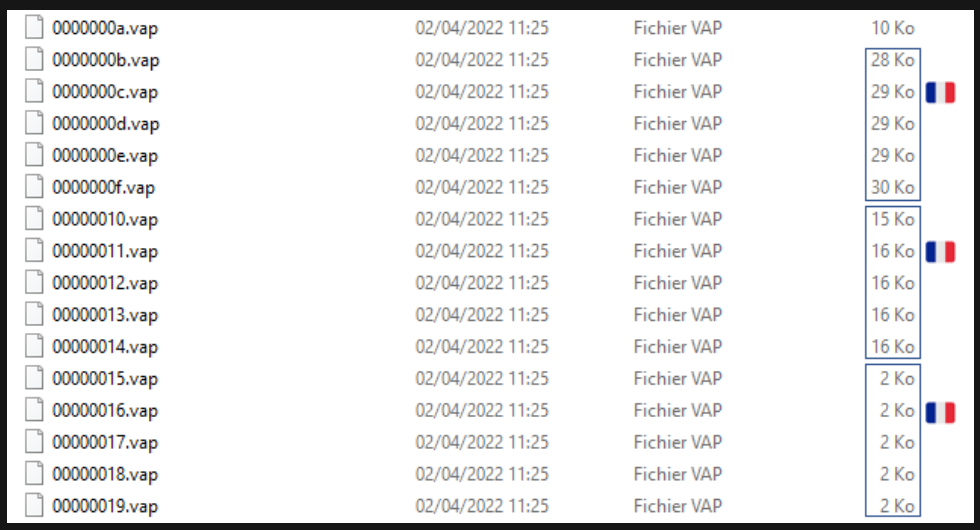
Que cherche-t-on ? Des textes, des images (textures) et parfois des modèles à traduire.

Méfiance avec les formats « conteneurs » comme ici le **.VAP** qui peut contenir différent types de fichiers, par exemple un ensemble de textes mélangés à une texture. Le **.DAT** peut aussi contenir différents formats, pour une autre raison… c'est que le script BMS donne cette extension par défaut lorsque le résultat de l'analyse du fichier n'est pas explicite.

Ensuite, avec un peu d'expérience on se rend vite compte que les **.G1T** sont des textures, c'est un cas fréquent (G1T, G1M, GT1…) dans les jeux publiées par Koei Tecmo.

Et en fait c'est à peu près tout, avec ces 3 formats, en plus des vidéos MP4 et du eboot.bin, on peut traduire le jeu dans son intégralité.

La version PS4 est un multi5 (En, Fr, Ge, Sp, It) et donc certains fichiers de la même taille se répètent forcément en 5 fois. Dans notre cas, ils ont même été extraits à la suite !



Ces VAP contenaient des textes et le français apparaissait toujours en 2ème position.

C’est encore plus flagrant dans le cas d’une image :

00000173.gt1	02/04/2022 11:25	Fichier GT1	16 385 Ko
00000174.gt1	02/04/2022 11:25	Fichier GT1	49 153 Ko
00000175.g1t	02/04/2022 11:25	Fichier G1T	49 153 Ko
00000176.gt1	02/04/2022 11:25	Fichier GT1	49 153 Ko
00000177.gt1	02/04/2022 11:25	Fichier GT1	49 153 Ko
00000178.gt1	02/04/2022 11:25	Fichier GT1	49 153 Ko
00000179.dat	02/04/2022 11:25	DAT	2 Ko

Ou dans le dossier « patch_data » qui contient les noms de fichiers avec le suffixe du pays (et donc le FR est 2ème place pour des raisons alphabétiques) :

Téléchargements › Dragon Quest Heroes II › PS4_extracted › uroot › patch_data › message

Nom	Modifié le	Type	Taille
P000_txtword_mvs_eng.bin	02/04/2022 01:09	Fichier BIN	
P000_txtword_mvs_fra.bin	02/04/2022 01:09	Fichier BIN	
P000_txtword_mvs_ger.bin	02/04/2022 01:09	Fichier BIN	
P000_txtword_mvs_ita.bin	02/04/2022 01:09	Fichier BIN	
P000_txtword_mvs_spa.bin	02/04/2022 01:09	Fichier BIN	

Le patch_data remplace certains fichiers compressés dans LINKDATA tout en indiquant explicitement leur nom, car il s’agit d’une mise à jour corrective sortie après le jeu, et donc au lieu de télécharger un gros .BIN de plusieurs Go, seule une partie du contenu est mise à jour, et ces fichiers ont la priorité sur la version de base. Ce qui donnera quelques doublons à repérer dans le Linkdata.

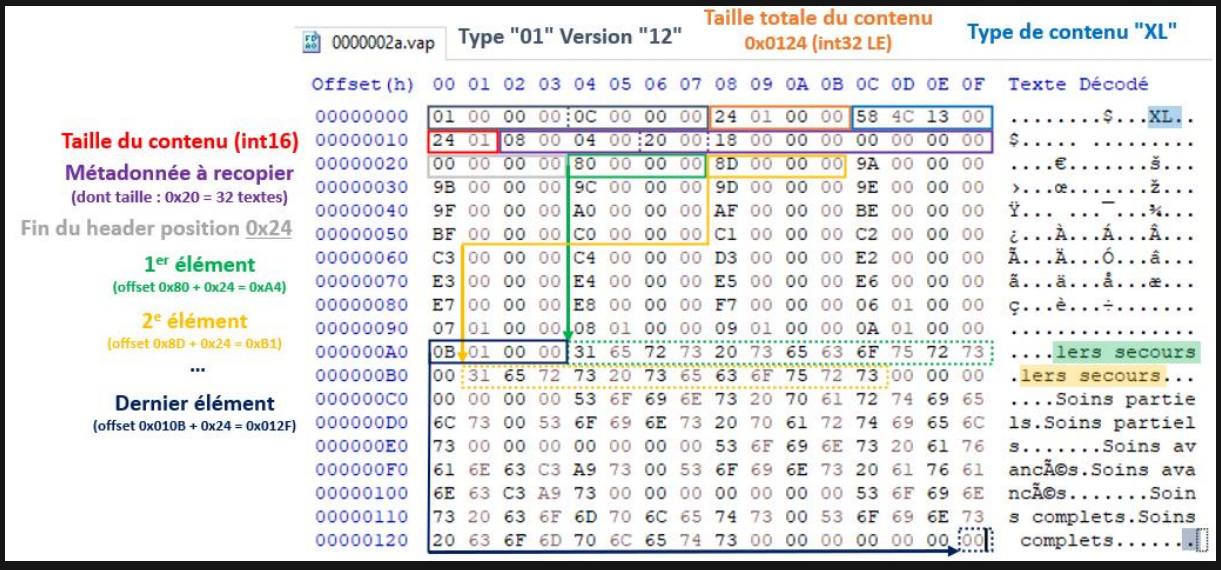
Donc, on récupère tout ça et au total on s’aperçoit que le projet consistera à traiter **~400 fichiers !**

À ce stade je me suis assuré aussi qu’il y aurait bien une méthode viable pour repack afin de ne pas perdre de temps inutilement sur quelque chose d’ingérable.

Modding des fichiers textuels

Des recherches approfondies pour savoir si un traducteur d’une autre nationalité aurait laissé ses outils librement sur internet n’ont rien donné. Le patch anglais de **Froid_san** est antérieur à la perte de son disque dur donc son code a été perdu. Les teams russes, souvent très actives dans le domaine, n’ont apparemment rien publié qui puisse aider. Enfin, les outils de modding d’autres jeux Koei Tecmo ne supportent malheureusement pas les .VAP extraits, par exemple ceux de Attack On Titan 2 sur Vita.

Bref, le format VAP doit être étudié pour extraction et insertion des textes, et ça se passe sur un éditeur hexadécimal. Voici le plus petit .VAP ouvert avec **HxD** et une déduction des segments de bytes :



Heureusement, ça reste assez simple et conventionnel. Le header contient des informations essentielles et on peut déterminer ce qui est constant entre les fichiers, et ce qui change. Il constitue les deux premières lignes de notre fichier hexa.

Ensuite une table indique le début de chaque "string" (chaîne de caractère) qui se lit en little-endian.

Et enfin les strings (encodage UTF-8 en français et Shift-JS en japonais) les unes à la suite des autres.

Cette analyse montre qu'il est possible :

- D'extraire toutes les lignes d'un fichier PS4 en français
- De copier le header du fichier PS Vita (japonais)
- De créer une table basée sur la longueur des textes français
- D'importer les textes français
- Et de mettre à jour le header avec la nouvelle taille de fichier

Ainsi c'est l'objectif de ce programme écrit en C# à partir de zéro connaissance en la matière :

extract_subs.cs ●

C: > Users > > Downloads > Dragon Quest Heroes II > Outils > Vap_oter > extract_subs.cs

```
30
31      static void Main()
32      {
33          List<dumpdata> OpenHead = HeaderGet();
34
35          if (File.Exists("out.txt"))
36          {
37              File.Delete("out.txt");
38          }
39
40      // Create a TXT file
41      using (StreamWriter sw = File.CreateText("out.txt"))
42      {
43          for (int i = 0; i < OpenHead.Count ; i++)
44          {
45              sw.WriteLine(OpenHead[i].NAMEEN);
46          }
47      }
48          //Unpack result
49          Console.WriteLine(" ======================================== ");
50          Console.WriteLine("NB : " + (OpenHead.Count - 1));
51
52          //Repack operations
53          if (1==1)    //    <------------------------------------------------------------
54          {
55              OpenHead = InjectTXT(OpenHead);
56              OpenHead = LineBreaks(OpenHead,50,2);
57              OpenHead = ReplaceAndFormat(OpenHead);
58              WriteVap(OpenHead,fileName);
59              UpdateHead(fileName + "_mod");
60              File.Move("in.txt","in_" + fileSource + ".txt");
61              File.Delete("out.txt");
62          }
```

C'est la fonction principale, les dérivées sont codées plus bas et utilisent les **BinaryReader/Writer**.

La fonction **HeaderGet** (L33) renvoie une liste d'objets « **dumpdata** » que j'ai défini contenant diverses informations, comme le texte, le numéro de ce texte, sa longueur en caractères, sa taille en octets…

Ensuite j'écris tous ces éléments obtenus dans un fichier « **out.txt** » et utilise diverses fonctions de repack peuvent être activées grâce à cette magnifique ligne de programmation **if(1==1)** que je remplace par **if(1==2)** pour les désactiver…

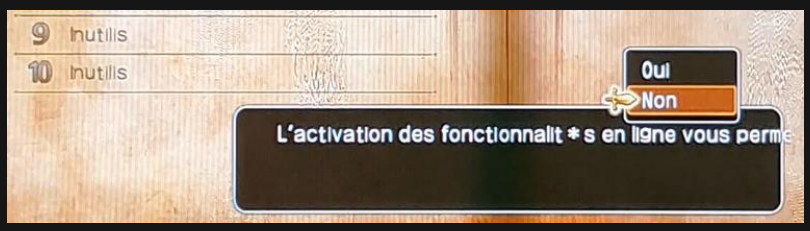
Parmi elles,

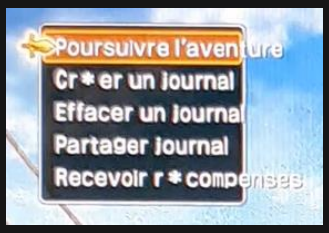
- **InjectTXT** (L55) permet de transformer un fichier « **in.txt** » en objet **dumpdata**
- **LineBreaks** (L56) de rajouter automatiquement des retours à la ligne
- **ReplaceAndFormat** (L57) d'effectuer des remplacements de caractères spéciaux et de réduire la longueur des textes
- **WriteVap** (L58) et **UpdateHead** (L59) de créer le fichier .vap modifié qui fonctionne sur PSVita

Merci aux forum C# et aux tutos qui m'ont permis de progresser ou d'utiliser des bouts de code.

Debug textes

Comme illustré ci-dessous, voici le résultat avec un remplacement brut des textes PS4 dans ceux PSVita





Et donc ça justifie bien ces fonctions de pré-traitement des textes avant injection, décrites juste avant : Selon la boîte de dialogue, il faut maîtriser la longueur des lignes, voire des mots ("Inutilisé" en 9 lettres ne rentre pas). Il faut aussi insérer les retours à la ligne explicitement. Les caractères spéciaux comme les accents français ne s'affichent pas.

On est chanceux, car dans certains jeux, une longueur de ligne qui dépasse la limite signifie un crash. Mais pas pour Dragon Quest Heroes 2, le pire scénario serait donc de ne pas voir la fin d'un message.

La partie qui consiste à traiter ce problème est la plus longue… il faut se motiver pendant des mois. On parle de la longueur en pixels et non pas de la longueur en nombre de lettres. J'aurais pu tenir compte du fait que « i » vaut 2 pixels, « m » 10 pixels et ainsi de suite pour tenter une approche automatisée, mais c'est une bonne chose de tout relire pour la qualité du patch.

Par conséquent me voici sur Excel en train de faire rentrer les textes dans les boîtes :

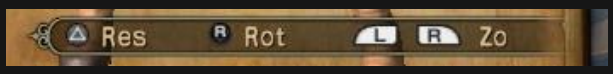
	D	E	F	L	M	N	O
1	Permets au vétéran chevronné que je suis de te donner quelques conseils, jeune guerrier !	戦いに　不慣れな貴様には@我輩が	Allow this grizzled veteran to offer you@some pointers, yo	##	41	26	90
828	Eh bien, tu étudieras toute seule ! Je serais¤vite diplômé et vivrais ma vie de chevalier !	俺は　士官学校を　首席で卒業す	Well, you'll be spending it on your own!@I'll be graduating as soon as I can, and@knighting my head off!	##	38	36	92
829	Ha ha ha ! Vous ne changerez jamais, hein ?	ぶっ……あはははははっ！@あんた	Ha ha ha ha! You never change, do you?	##	41	26	44

La largeur de la colonne D me donne une indication de l'endroit où sera le retour à la ligne sur la PSVita. En rouge le nombre de caractères s'il dépasse 85. Ici 2 lignes sont autorisées et entre 40 et 45 caractères maximum par ligne. Enfin, j'utilise ¤ ou @ pour forcer les retours à la ligne lorsqu'il y a un risque que le saut automatique en C# ne soit pas bien placé.

Ainsi sont passée à peu près 3200 dialogues sous le joug de ma vérification minutieuse !

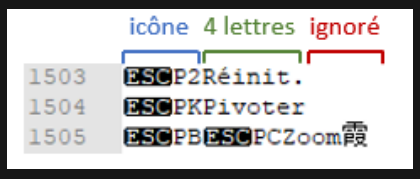
Le japonais peut exprimer des mots assez longs en seulement quelques caractères… Que faire lorsque la limite est vraiment trop contraignante ? Changer de mot, une abréviation, ou alors une 3ème technique. Tout le monde se rappelle de l'abréviation « $^{P}_{K}M_{N}$ », et bien il n'y avait la place que pour 3 symboles afin d'écrire « changer de pokémon » et donc la font (police de caractères) a été éditée pour intégrer 2 symboles : [$^{P}_{K}$] et [$^{M}_{N}$].

Cette dernière méthode est aussi la solution pour insérer des accents.



Ci-dessus comparaison du patch FR, du jeu japonais et du patch anglais. 4 caractères sont autorisés.

Pour réussir à écrire des mots plus longs, le script et la font (tuiles de 20x20 px) ont été modifiés :



Il est ainsi possible d'écrire R éi nit . / Pi vo te r et Zo om ± dans la limite de 4 caractères !

Tous les remplacements se font ensuite automatiquement avec le code C#. L'inconvénient est qu'on ne peut pas mettre 1 seul accent dans 20 pixels, sous peine d'obtenir un résultat « ab î m é » avec des espaces moches dans les mots. Donc en fait, toutes les combinaisons de 2 accents sont insérées dans la font, et on a encore de la marge pour en rajouter… (seulement ¼ est montré sur cette capture) :

En usant et abusant de cette méthode, son pouvoir de réduction des phrases/mots devient de plus en plus intéressant, comme nous sommes limités en nombre de caractères affichés à l'écran.

J'ai donc brièvement travaillé sur la fréquence d'occurrence des doublets de lettres dans la langue française, afin de généraliser la réduction de taille sur tous mes scripts.

On trouve une liste de 1500 mots sur le [ministère de l'éducation française] :

> Une liste rassemblant près de 1500 mots, les plus fréquents de la langue française, a été constituée par le lexicologue Étienne Brunet. Elle rend compte de la langue que lisent les élèves francophones. Ces mots, extraits de textes littéraires ou non, ont été ramenés à leur base lexicale.

En la décomposant par couples de 2 lettres, tout en gardant la pondération de chaque mot, il est possible de résumer en 377 doublets classés par fréquence d'apparition dans la langue française !

Mot	Occurrence	Doublet	Occurrence
le	1050561	le	1646215
de	862100	re	1380555
un	419564	de	1211002
être	351960	er	963019
et	362093	ir	905090

Résultat : On peut constater que le mot le plus fréquent est bien évidemment « le/la/les » et que le top-5 des doublets de lettres les plus fréquents est constitué de « le », « re », « de », « er » et « ir ».

Mais surtout en intégrant ces 377 tuiles dans mon fichier de police de caractères, je réduis jusqu'à 50% la longueur des mots. En rajoutant les accents, la ponctuation, les majuscules, quelques symboles (comme « 1er ») le total fait **plus de 800** caractères intégrés dans la font pour un rendu optimal.

Exemples qui ont pu être ainsi corrigés :

Enfin, le principal travail de beta-testing consistait à vérifier que tout rentrait bien comme il faut sans problème d'affichage, certains cas restant malheureusement impossibles.

Lorsque les scripts et la font sont perfectionnés, je peux lancer le programme C# de création des fichiers moddés et ainsi faire en sorte que toutes les longueurs, nombres de lignes, et caractères spéciaux soient intégrés proprement.

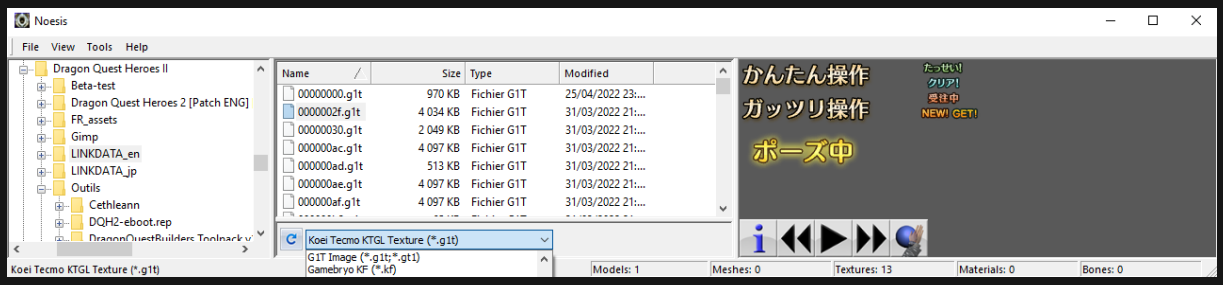
```
 ========================================
Source found : _TO_PATCH/in_P005_txtsentence01_3x50.txt
Base found : Source files/P005_txtsentence01_jp.bin
Nfiles : 1600
 ========================================
Lines max : 3
Chars max : 50
Size : 107741 / 1A4DD
 > Patch SUCCESS !
 ========================================
Source found : _TO_PATCH/in_P206_char_nms_t03_1x50.txt
Base found : Source files/P206_char_nms_t03_jp.bin
Nfiles : 420
 ========================================
Lines max : 1
Chars max : 50
Size : 4307 / 10D3
 > Patch SUCCESS !

[Done] exited with code=0 in 3.582 seconds
```

Modding des fichiers graphiques

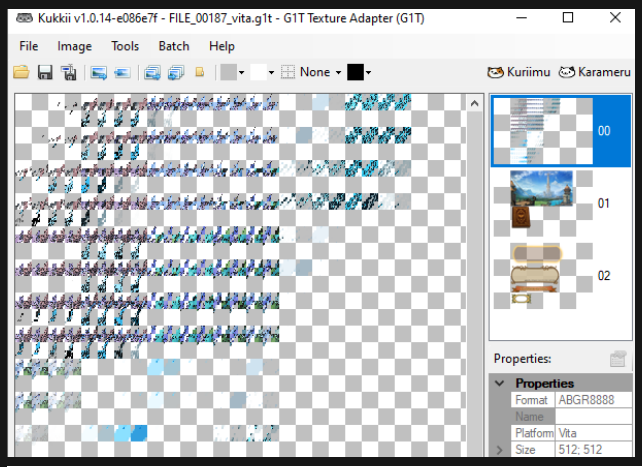
Si l'édition des textes consiste la majeure partie d'un projet de traduction, la manipulation des « textures » (images utilisées par le jeu) est néanmoins une partie non négligeable et parfois difficile.

Les types de fichiers varient en fonction du jeu, du développeur du jeu, et de la plateforme. La majorité de ce qui concerne DQH2 est sous forme de textures G1T et de conteneurs BIN/VAP.

Ce format n'est pas inconnu et certains outils ont été développés pour les extraire, mais les images peuvent être codées de différentes manières (DXT1, DXT3, RGBA8, …) et pour la PSVita les pixels sont mélangés (swizzled) pour être lus dans un ordre particulier, ce qui rend parfois la tâche difficile.



Noesis permet seulement d'exporter certains G1T en PNG. Ayant déjà trouvé une astuce pour importer les textures G1T dans mon patch de Attack on Titans 2, développé par le même studio, je ne pars pas de zéro ! En effet le logiciel Kukkii.exe de la suite Kuriimu (la version 1, pas la nouvelle) permet d'éditer des textures si leur nom de fichier se termine par « _vita », ce qui indique un swizzle.



Cependant, la plupart des textures sont constituées de plusieurs images et **kukkii** ne les gère pas toutes, il faut donc s'appuyer sur un deuxième outil nommé **Gust_tools**.

Il gère beaucoup de formats et par chance nos images manquantes sont reconnues comme ABGR et ARGB ! Lorsqu'on les extrait, le seul problème est que la texture obtenue a ses canaux bleu « B » et rouge « R » inversés. Donc il faut effectuer la rotation.

0de9af9ff9 gust_tools / gust_g1t.c

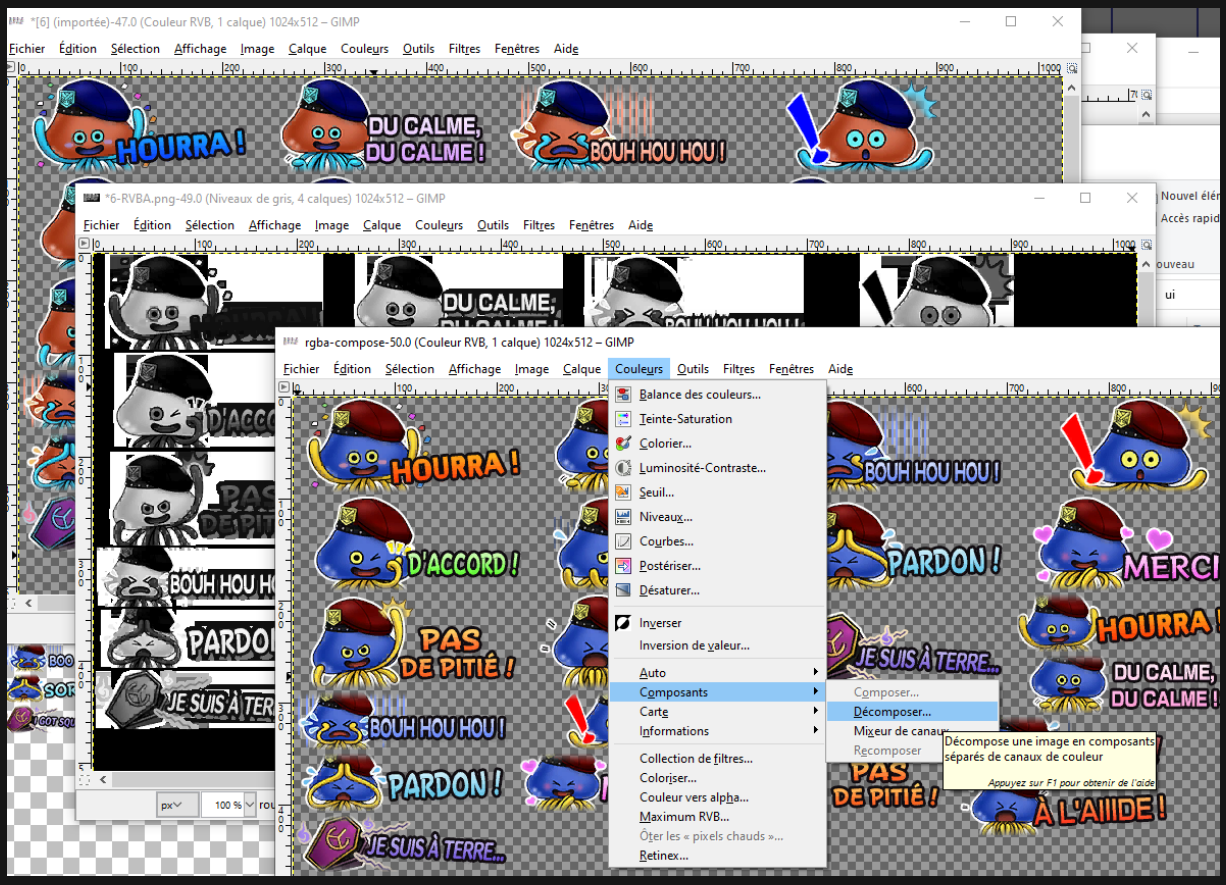
VitaSmith gust_g1t: fix regression for BC7 and add SRGB support ...

1 contributor

794 lines (729 sloc) 34.3 KB

```
654             case 0x00: texture_format = DDS_FORMAT_ABGR; bits_per_pixel = 32; break;
655             // Note: Type 0x01 may also be DDS_FORMAT_ARGB for PS Vita images
656             case 0x01: texture_format = DDS_FORMAT_RGBA; bits_per_pixel = 32; break;
657             case 0x06: texture_format = DDS_FORMAT_DXT1; bits_per_pixel = 4; break;
658             case 0x07: texture_format = DDS_FORMAT_DXT3; bits_per_pixel = 8; supported = false; break;  // UNSUPPORTED!!
659             case 0x08: texture_format = DDS_FORMAT_DXT5; bits_per_pixel = 8; break;
660             case 0x09: texture_format = DDS_FORMAT_GRAB; bits_per_pixel = 32; break;
661             case 0x10: texture_format = DDS_FORMAT_DXT1; bits_per_pixel = 4; supported = false; break;  // UNSUPPORTED!!
662             case 0x12: texture_format = DDS_FORMAT_DXT5; bits_per_pixel = 8; supported = false; break;  // UNSUPPORTED!!
663             case 0x21: texture_format = DDS_FORMAT_ARGB; bits_per_pixel = 32; break;
664             case 0x3C: texture_format = DDS_FORMAT_DXT1; bits_per_pixel = 16; supported = false; break; // UNSUPPORTED!!
665             case 0x3D: texture_format = DDS_FORMAT_DXT1; bits_per_pixel = 16; supported = false; break; // UNSUPPORTED!!
666             case 0x45: texture_format = DDS_FORMAT_BGR; bits_per_pixel = 24; break;
667             case 0x59: texture_format = DDS_FORMAT_DXT1; bits_per_pixel = 4; break;
668             case 0x5B: texture_format = DDS_FORMAT_DXT5; bits_per_pixel = 8; break;
669             case 0x5C: texture_format = DDS_FORMAT_BC4; bits_per_pixel = 4; break;
```

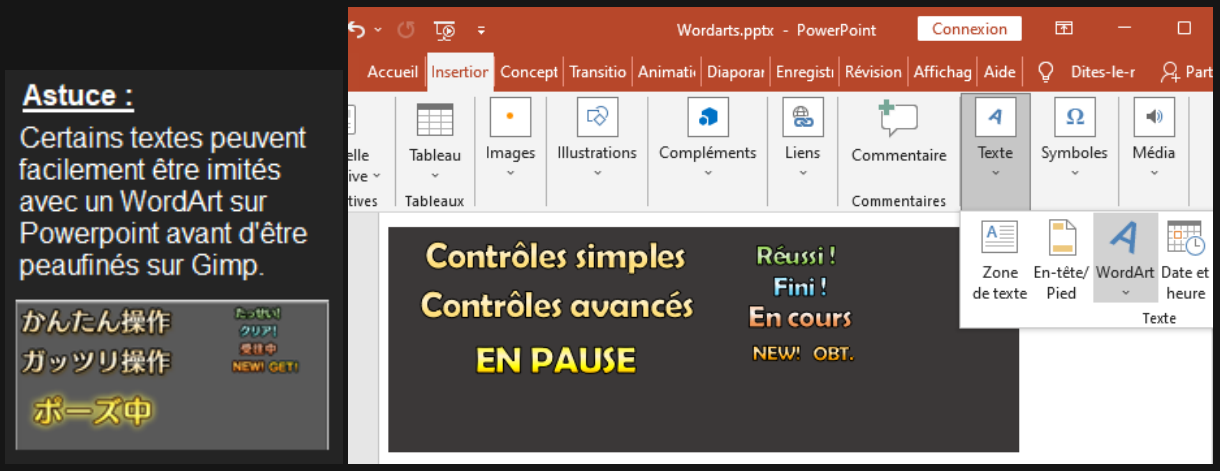
Gimp est l'outil parfait pour modifier des images, il ouvre les DDS sans problème et permet d'exporter au format souhaité et décomposer / recomposer les images.



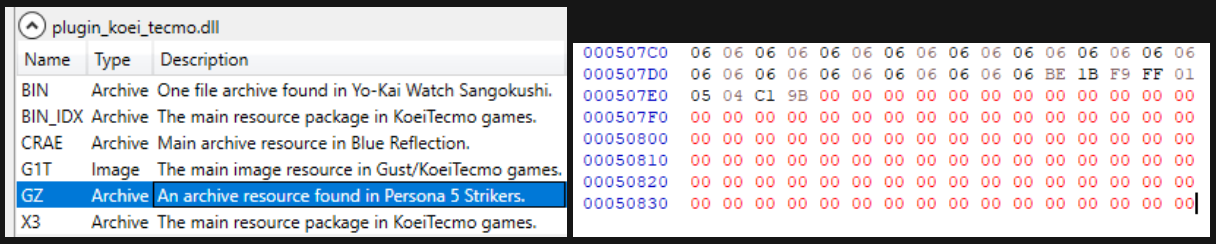
<u>Enfin, pour terminer la partie sur les textures, quelques exemples :</u>

Les sprites PS4 ne sont pas aux mêmes coordonnées que la version Vita ! C'était trop simple…

Certaines traductions étaient introuvables dans la version PS4 mais ont été reproduites.



Certains fichiers étaient compressés sous format G1T.GZ. Kuriimu2 permet d'effectuer l'opération mais il faut manuellement rajouter des zéros à la fin du fichier. C'est con mais il m'a fallu pas mal de temps pour le deviner…



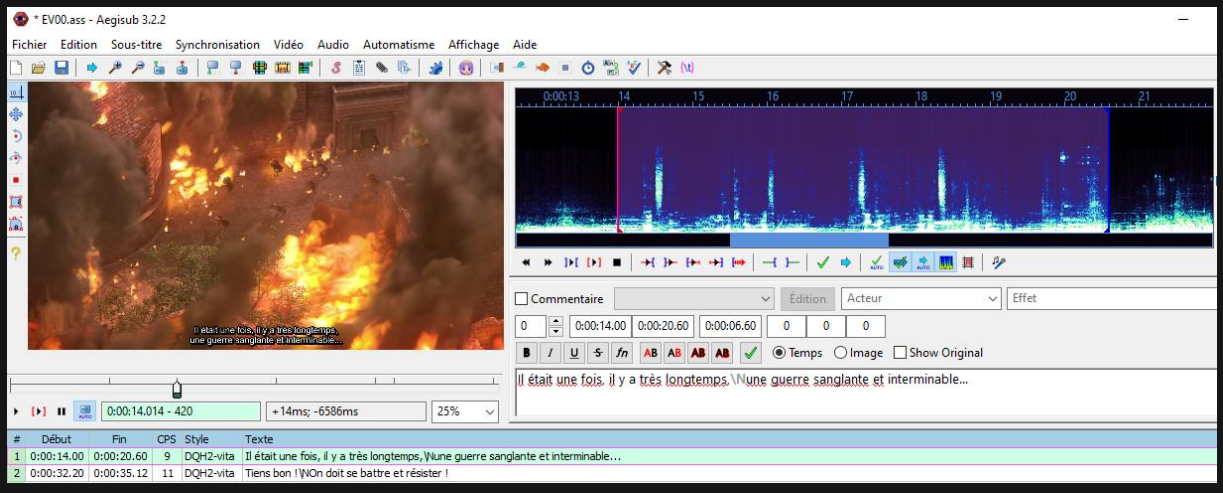
Ainsi, une 40aine de textures ont été intégrées avec succès dans le patch !

Modding des vidéos

Il y avait un fichier de sous-titres dans les assets de la version PS4, mais une fois importé sur Vita l'affichage des caractères sur la vidéo était atroce, pas du tout adapté à la longueur des phrases, décalé sur l'écran. Donc je l'ai remplacé par un fichier vide, et décidé d'incruster les ST sur les vidéos. Et justement, elles sont en MP4 ce qui est très pratique pour les manipuler.

Cette opération prend du temps malheureusement, car il faut « timer » toutes les lignes (leur donner un temps de début et de fin) et qu'il y a beaucoup de vidéos.

2000 lignes ont été traitées avec Aegisub pour 45 vidéos, ce qui a pris plusieurs semaines à faire.

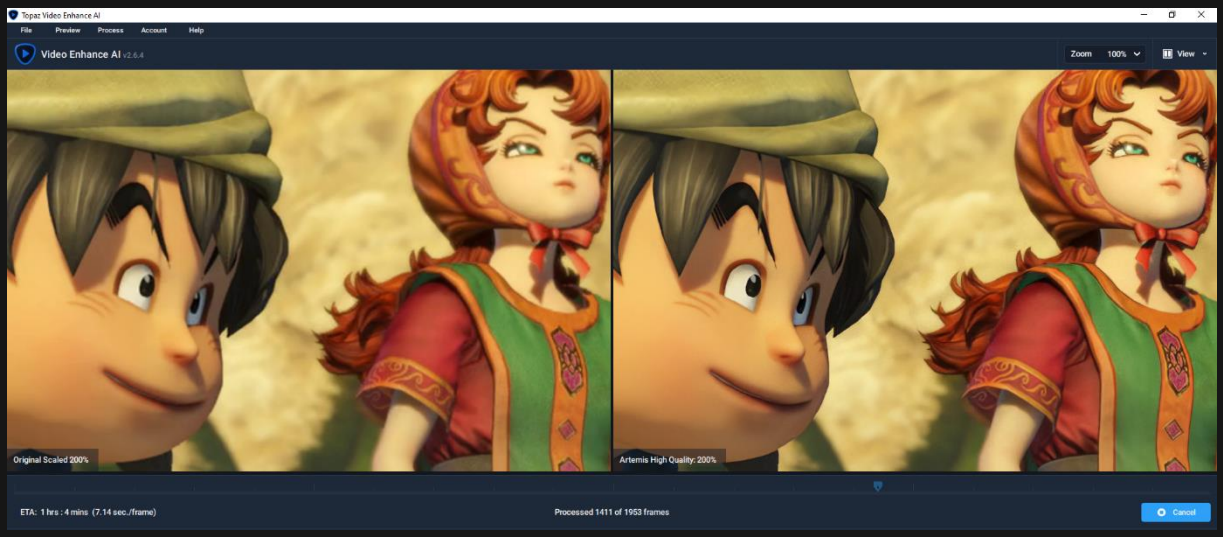


Ensuite, il faut passer par un logiciel encodeur de vidéos pour fusionner les ST et l'image.

Quitte à ré-encoder les vidéos, je suis parti de celles sur PS4 qui ont une meilleure résolution et j'ai gardé une qualité supérieure à celle qui a été appliquée sur la version PSVita, donc aussi augmenté la taille des vidéos (bitrate + élevé).

Mais en voulant faire cette amélioration « HQ » j'ai vu qu'il manquait 1 vidéo sur PS4, et donc je suis passé par un process de « upscale » (augmentation de la résolution), avec des outils modernes basées sur de l'IA, pour ensuite la reconvertir à l'échelle PSVita en ajoutant les sous-titres.

Topaz Video Enhance AI est vraiment surprenant par la qualité du résultat



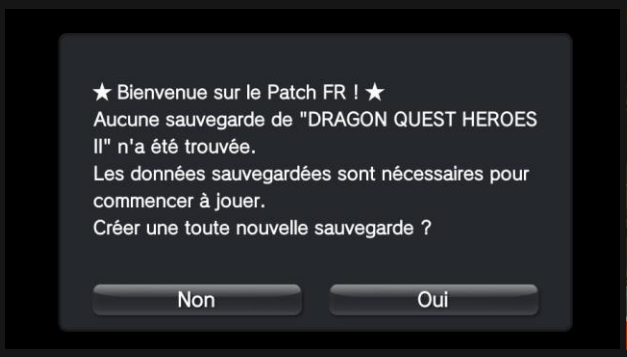
Le dossier « movie » est donc passé de 2 Go à 3 Go mais avec une amélioration nette de la qualité !

Le eboot.bin

Il s'agit du programme principal du jeu, celui qui va initialiser, paramétrer, commander tout ce qu'il se passe lorsque la Vita exécute DQH2.

Une décompilation en langage C de ce fichier .ELF est possible (avec Ghidra par ex.) et offrirait d'énormes possibilités de modding. Cependant elle est ultra-complexe et hors de mes capacités, donc je me contente de regarder s'il n'y a pas des fichiers à traduire contenus dans cet EBOOT.

Donc avec les outils de reverse PSVITA, un coup de vita-unmake-self pour déchiffrer et vita-elf-inject pour injecter seront suffisant pour notre affaire. L'EBOOT modifié traduit le clavier en mode européen ainsi que quelques messages système :



Les echecs

Même après avoir passé en revue des dizaines de fois tous les fichiers, certaines textures récalcitrantes n'ont jamais voulu se montrer ! Il y a deux traductions qui manquent toujours à l'appel :

Même en ayant trouvé la texture qui fait l'effet flamme, impossible de trouver ce ジャイワー. . .

Le titre !! C'est tellement frustrant de savoir que 99,9% du jeu est traduit, mais que le 0,1% restant se voit comme le nez au milieu de la figure lorsqu'on démarre le jeu…

Le nom japonais est : **Dragon Quest Heroes II : Les rois jumeaux et la fin de la prophétie**

Tant pis pour ces deux-là, j'abandonne.

Conception du patch

J'ai l'habitude de fournir les fichiers déjà patchés. La plupart du temps un patcheur différentiel n'est pas adapté car les fichiers d'origine sont cryptés et n'ont rien en commun avec ceux moddés. D'autant plus qu'avec RePatch, le contenu partagé n'est pas une ROM complète.

Brièvement, j'évoque ici la manière de reconstruire le LINKDATA.BIN et LINKDATA.IDX. La solution a été donnée par Falo ici : https://gbatemp.net/threads/dragon-quest-builders-2.528161/

Avec un fichier batch il est possible d'ajouter les fichiers moddés à la fin du BIN et de modifier le IDX.

```
4   linkdata.exe inject "%DIR%\FILE_00005.dat" 005 "LINKDATA.IDX"
5   linkdata.exe inject "%DIR%\FILE_00007.dat" 007 "LINKDATA.IDX"
6   linkdata.exe inject "%DIR%\FILE_00008.dat" 008 "LINKDATA.IDX"
```

Donc : Enfin, on rassemble tous les fichiers édités précédemment dans un dossier repatch, on fait un beau paquet-cadeau, on écrit un tutoriel et on publie !

Mot de la fin

Ce projet totalise très exactement 5 mois, entre Mars et Juillet 2022. Il a été accompli par une seule personne (moi, s'il fallait préciser) et n'était pas forcément attendu par beaucoup. Sans faire une béta-test complète, je peux tout de même constater que le résultat atteint mes exigences de qualité. Mais j'aime bidouiller sur PSVita plus que de jouer sur PSVita, et je n'ai pas un intérêt particulier pour ce jeu.

Deuxièmement, j'aime aussi atteindre mes objectifs. Et lorsque je m'engage dans une tache, j'ai le sentiment de devoir la finir pour ne pas décevoir ceux qui en attendent le résultat, ou qui m'ont aidé. C'est pour cela que je ne peux pas m'engager sur une traduction complète, mais que je sélectionne des projets où la traduction FR est disponible sur une autre plateforme que la PS Vita.

Et enfin, j'aime aussi partager sur la scène PS Vita ! Donc lorsque l'on a passé 5% de fun et 95% de travail acharné, ennuyeux, répétitif ou frustrant sur son temps libre… le moment de publier le patch est autant un aboutissement qu'une libération !

Satisfait également d'avoir pu écrire ce making-off, afin que vous puissiez constater que même un portage de traduction n'est pas trivial en quantité de travail, ou pour susciter quelques vocations en démontrant que la difficulté n'est pas si élevée.

En tous les cas, je suis super fier de vous présenter à nouveau un de mes patchs FR sur PS Vita et j'espère que vous l'apprécierez.

- Firo

alias Maxscr